Amendments to the Claims:

This listing of claims will replace all prior versions, and listing, of claims in the application:

Listing of Claims:

1. (currently amended) A system comprising:

a plurality of arbiters that each simultaneously arbitrate among common elements of a ~~common~~ resource; and

conflict logic configured to detect conflicts among the plurality of arbiters accessing the elements of the ~~common~~ resource, and, when a conflict is detected, the conflict logic is configured to alter processing relating to the conflict in one of the conflicting arbiters.

2. (currently amended) The system of claim 1, wherein the ~~common~~ resource [[is]] includes a set of queues and the elements of the ~~common~~ resource are individual ones of the queues within the set.

3. (currently amended) The system of claim 1, wherein the plurality of arbiters ~~includes~~ include a first arbiter and a second arbiter, the first and second arbiter each being implemented as a series of pipeline stages.

4. (currently amended) The system of claim 3, wherein the first arbiter arbitrates based on flow control and the second arbiter arbitrates to manage

congestion of the elements in the ~~common~~ resource, and wherein the first arbiter has a higher priority than the second arbiter.

5. (original) The system of claim 1, wherein the conflict logic alters the processing relating to the conflict in the one of the conflicting arbiters when the one of the conflicting arbiters has not passed a predetermined point in its processing.

6. (currently amended) The system of claim 5, wherein the predetermined point is a predetermined stage in [[the]] a pipeline of the second arbiter.

7. (currently amended) The system of claim 1, additionally comprising:

logic configured to, when the conflict logic detects a conflict between the plurality of arbiters and the one of the conflicting arbiters has passed a predetermined point in processing, modify the element associated with the conflict such that the higher priority arbiter is immediately able to access a next data element in the ~~common~~ resource.

8. (currently amended) The system of claim 7, wherein the element of the ~~common~~ resource is a first-in-first-out (FIFO) queue and the logic advances a head pointer of the FIFO queue to point to the next data element.

9. (currently amended) A method comprising:

examining a plurality of arbiters that arbitrate among a plurality of queues for conflicts among the plurality of arbiters in arbitrating the plurality of queues;

determining, when conflicts occur in arbitrating the plurality of queues, whether one of the conflicting arbiters has reached an arbitration point beyond a predetermined commit point; and

invalidating processing in the one arbiter related to the conflict when the one arbiter is not beyond the commit point.

10. (original) The method of claim 9, further comprising:

modifying the queue associated with the conflict so that a next data item in the queue is advanced to a head position in the queue when the lower priority arbiter is beyond the commit point.

11. (original) The method of claim 9, wherein another arbiter receives the next data item when the one arbiter is beyond the commit point.

12. (original) The method of claim 9, further comprising, when the one arbiter is beyond the commit point and the queue does not contain a next data item:

invalidating processing in the another arbiter relating to the conflict.

13. (original) The method of claim 9, wherein the plurality of arbiters includes a first arbiter and a second arbiter, the first and second arbiter each being implemented as a series of pipelined stages.

14. (currently amended) The method of claim 13, wherein the first arbiter arbitrates based on flow control and the second arbiter arbitrates among the ~~data items~~ queues to manage congestion in the common resource, and wherein the first arbiter has a higher priority than the second arbiter.

15. (original) The method of claim 13, wherein the commit point is a predetermined stage in the pipeline of the second arbiter.

16. (original) The method of claim 9, wherein the plurality of queues are each first-in-first-out (FIFO) queues.

17. (original) A device comprising:

a plurality of queues;

a first arbiter configured to select from among the plurality of queues and to receive data items from the selected queue;

a second arbiter configured to monitor the plurality of queues for congestion and to drop data items from congested queues; and

conflict detection logic coupled to the plurality of queues, the first arbiter, and the second arbiter, the conflict detection logic detecting conflicts between the

first and second arbiters in arbitrating the plurality of queues, and, when a conflict is detected, altering processing relating to the conflict in the second arbiter when the second arbiter has not passed a predetermined commit point in processing a queue.

18. (original) The device of claim 17, wherein the first and second arbiters are implemented as a series of pipelined stages.

19. (original) The device of claim 18, wherein the commit point is a predetermined stage in the second arbiter.

20. (original) The device of claim 17, additionally comprising:

logic configured to, when the conflict logic detects a conflict between the first and second arbiter and the second arbiter has passed the commit point, modify the queue associated with the conflict such that the first arbiter is immediately able to access a next data item in the queue associated with the conflict.

21. (original) The device of claim 20, wherein the logic advances a head pointer of the queue associated with the conflict to point to the next data item in the queue.

22. (currently amended) A network device comprising:

a plurality of processing elements, the processing elements transmitting data items to one another and transmitting the data items to destinations external to the network device, the processing elements including

a plurality of queues configured to store the data items before transmission of the data items,

a plurality of arbiters that independently arbitrate among data items in the queues, and

conflict logic configured to detect conflicts among the plurality of arbiters in accessing the queues, and, when a conflict is detected, the conflict logic is configured to clear processing relating to the conflict in one of the conflicting arbiters when the one of the conflicting arbiters has not passed a predetermined commit point.

23. (original) The network device of claim 22, wherein the network device is a router.

24. (original) The network device of claim 22, wherein the plurality of arbiters includes a first arbiter and a second arbiter, the first and second arbiter being implemented as a series of pipeline stages.

25. (original) The network device of claim 24, wherein the first arbiter arbitrates based on flow control and the second arbiter arbitrates to manage congestion in the queues.

26. (original) The network device of claim 22, wherein the commit point is a predetermined stage in the pipeline of the one arbiter.

27. (original) The network device of claim 22, additionally comprising:

logic configured to, when the conflict logic detects a conflict between the plurality of arbiters and the one of the conflicting arbiters has passed the commit point, modify the queue associated with the conflict such that the other arbiter is immediately able to access a next data item in the queue.

28. (original) A device comprising:

means for detecting conflicts among a plurality of arbiters that arbitrate among a plurality of queues;

means for determining, when conflicts are detected by the means for detecting, whether one of the conflicting arbiters has reached an arbitration point beyond a predetermined commit point; and

means for invalidating processing relating to the conflict in the one arbiter when the one arbiter is not beyond the commit point.

29. (original) The device of claim 28, further comprising:

means for modifying the queue associated with the conflict so that a next data element in the queue is advanced to a head position in the queue when the one arbiter is beyond the commit point.